Predicting Lagrangian Multipliers for Mixed Integer Linear Programs

Francesco Demelas [1] **Joseph Le Roux** [1] **Mathieu Lacroix** [1] **Axel Parmentier** [2]

Abstract

Lagrangian Relaxation stands among the most efficient approaches for solving Mixed Integer Linear Programs (MILPs) with difficult constraints. Given any duals for these constraints, called Lagrangian Multipliers (LMs), it returns a bound on the optimal value of the MILP, and Lagrangian methods seek the LMs giving the best such bound. But these methods generally rely on iterative algorithms resembling gradient descent to maximize the concave piecewise linear dual function: the computational burden grows quickly with the number of relaxed constraints. We introduce a deep learning approach that bypasses the descent, effectively amortizing *per instance* optimization. A probabilistic encoder based on a graph neural network computes, given a MILP instance and its Continuous Relaxation (CR) solution, high-dimensional representations of relaxed constraints, which are turned into LMs by a decoder. We train the encoder and the decoder jointly by directly optimizing the bound obtained from the predicted multipliers. Our method is applicable to any problem with a compact MILP formulation, and to any Lagrangian Relaxation providing a tighter bound than CR. Experiments on two widely known problems, Multi-Commodity Network Design and Generalized Assignment, show that our approach closes up to 85 % of the gap between the continuous relaxation and the best Lagrangian bound, and provides a high-quality warm-start for descent-based Lagrangian methods.

[1]Laboratoire d'Informatique de Paris-Nord, Université Sorbonne Paris Nord — CNRS, France [2]CERMICS, École des Ponts, France. Correspondence to: Francesco Demelas <demelas@lipn.fr>, Joseph Le Roux <leroux@lipn.fr>, Mathieu Lacroix <lacroix@lipn.fr>.

Proceedings of the 41^{st} International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

1. Introduction

Mixed Integer Linear Programs (MILPs) (Wolsey, 2021) have two main strengths that make them ubiquitous in combinatorial optimization (Korte & Vygen, 2012). First, they can model many combinatorial optimization problems. Second, extremely efficient solvers can now handle MILPs with millions of constraints and variables. They therefore have a wide variety of applications in logistics, telecommunications and beyond. MILP algorithms are exact: they return an optimal solution, or an optimality gap between the returned solution and an optimal one.

MILPs are sometimes hard to solve due to a collection of difficult constraints. Typically, a small number of constraints link together otherwise independent subproblems. For instance, in vehicle routing problems (Golden et al., 2008), there is one independent problem for each vehicle, except for the linking constraints that ensure that exactly one vehicle operates each task of interest. Lagrangian relaxation approaches are popular in such settings as they allow to unlink the different subproblems.

More formally (Conforti et al., 2014, Chap. 8), let P be a MILP of the form:

$$
\begin{aligned}
(P) \qquad & \min_{\boldsymbol{x}} \ \boldsymbol{w}^\top \boldsymbol{x} && \text{(1a)}\\
& \boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b} && \text{(1b)}\\
& \boldsymbol{C}\boldsymbol{x} \geq \boldsymbol{d} && \text{(1c)}\\
& \boldsymbol{x} \in \mathbb{R}_+^m \times \mathbb{N}^p && \text{(1d)}
\end{aligned}
$$

While CR amounts to simply removing the integrity constraints (*i.e.* (1d) becomes $\boldsymbol{x} \in \mathbb{R}_+^{m+p}$), the relaxed Lagrangian problem is obtained by dualizing difficult constraints (1b) and penalizing their violation with Lagrangian multipliers (LMs) $\boldsymbol{\pi} \geq \mathbf{0}$:

$$
\begin{aligned}
\big(LR(\boldsymbol{\pi})\big) \qquad & \min_{\boldsymbol{x}} \boldsymbol{w}^\top \boldsymbol{x} + \boldsymbol{\pi}^\top (\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x})\\
& \boldsymbol{C}\boldsymbol{x} \geq \boldsymbol{d}\\
& \boldsymbol{x} \in \mathbb{R}_+^m \times \mathbb{N}^p
\end{aligned}
$$

Standard weak Lagrangian duality ensures that $LR(\boldsymbol{\pi})$ is a lower bound on P. The Lagrangian dual problem aims at

finding the best such bound:

$$(LD) \qquad \max_{\boldsymbol{\pi} \geq \mathbf{0}} LR(\boldsymbol{\pi}).$$

Geoffrion's theorem (1974) ensures that LD is a lower bound at least as tight as the continuous relaxation. It is strictly better on most applications. Beyond this bound, Lagrangian approaches are also useful to find good primal solutions. Indeed, Lagrangian heuristics (Beasley, 1990) exploit the dual solution $\boldsymbol{\pi}$ and the variable assignment for $\boldsymbol{x}$ of $LR(\boldsymbol{\pi})$ to compute good quality solutions of (1a)-(1d). Note that both the bound and the heuristic hold even in the case of non-optimal duals $\boldsymbol{\pi}$. We define *good* Lagrangian duals $\boldsymbol{\pi}$ as those that lead to a bound $LR(\boldsymbol{\pi})$ better than the CR solution, and thus closer to LD.

Since $\boldsymbol{\pi} \mapsto LR(\boldsymbol{\pi})$ is piecewise linear and concave, it is generally optimized using a subgradient algorithm. Unfortunately, the number of iterations required to obtain good duals quickly increases with the dimension of $\boldsymbol{\pi}$, which makes the approach extremely intensive computationally.

In this work[1] we introduce a state-of-the-art encoder-decoder neural network that computes *good* duals $\boldsymbol{\pi}$ from the CR solution. The probabilistic encoder $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\iota)$, based on a graph neural network (GNN), takes as input a MILP instance ι as well as the primal and dual CR solutions, and returns an embedding of the instance, where each dualized constraint is mapped to a high-dimensional dense vector. The deterministic decoder $f_{\boldsymbol{\theta}}(\boldsymbol{z})$ reconstructs single dimensional duals from constraint vectors. The learning objective is unsupervised since the Lagrangian dual function $LR(\boldsymbol{\pi})$ leads to a natural loss function that does not require gold references. Experiments on two standard and widely used problems from the Combinatorial Optimization literature, Multi-Commodity Network Design and General Assignment, show that the predicted duals close up to 85% of the gap between the CR and LD solutions. Finally, when optimal duals are the target, we show that predicted duals provide an excellent warm-start for state-of-the-art descent-based algorithms for objective (1). Our approach is restricted to compact MILPs and Lagrangian Relaxations admitting a tighter bound than CR since primal and dual CR solutions are part of the GNN input.

2. Learning Framework

2.1. Overall Architecture

Iterative algorithms for setting LMs to optimality such as the subgradient method (SM) (Polyak, 1987, Chap 5.3) or the Bundle method (BM) (Hiriart-Urruty & Lemaréchal, 1996; Le et al., 2007) start by initializing LMs. They can be set to zero but a solution considered as better in practice by the Combinatorial Optimization community is to take advantage of the bound given by CR and its dual solution, often computationally cheap for compact MILPs. Specifically, optimal values of the CR dual variables identified with the constraints dualized in the Lagrangian relaxation can be understood as LMs. In many problems of interest these LMs are not optimal and can be improved by SM or BM. We leverage this observation by trying to predict a deviation from the LMs corresponding to the CR dual solution.

The architecture is depicted in Figure 1. We start from an input instance ι of MILP P with a set of constraints for which the Lagrangian relaxed problem is easy to compute, then solve CR and obtain the corresponding primal and dual solutions. The input enriched with CR solutions is then passed through a probabilistic encoder, composed of three parts: *(i)* the input is encoded as a bipartite graph in a way similar to (Gasse et al., 2019), also known as a *factor graph* in probabilistic modelling, and initial graph node feature extraction is performed, *(ii)* this graph is fed to a GNN in charge of refining the node features by taking into account the structure of the MILP, *(iii)* the last layer of the GNN is used to parameterize a distribution from which vectors $\boldsymbol{z}_c$ can be sampled for each dualized constraint c.

The decoder then translates $\boldsymbol{z}_c$ to a positive LM $\pi_c = \lambda_c + \delta_c$ by predicting a deviation δ_c from the CR dual solution variable λ_c. Finally, the predicted LMs can be used in several ways, in particular to compute a Lagrangian bound or to warm-start an iterative solver.

2.2. Objective

We train the network's parameters in an end-to-end fashion by maximizing the average Lagrangian bound $LR(\boldsymbol{\pi})$ obtained from the predicted LMs $\boldsymbol{\pi}$ over a training set. This can be cast as an empirical risk optimization, or an Energy-Based Model (Le Cun et al., 2006) with latent variables, where the Lagrangian bound is the (negative) energy corresponding to the coupling of the instance with the subproblem solutions, and the LMs — or more precisely their high-dimensional representations — the latent variables. For our problem, a natural measure of the quality of the prediction is provided by the value LR that we want to maximize to tighten the duality gap. Given an instance ι we want to learn to predict the latent representations $\boldsymbol{z}$ of the LMs for which the Lagrangian bound is the highest:

$$\max_{\boldsymbol{\phi},\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{z} \sim q_{\boldsymbol{\phi}}(\cdot|\iota)} \left[LR([\boldsymbol{\lambda} + f_{\boldsymbol{\theta}}(\boldsymbol{z})]_+; \iota) \right]$$

where $q_{\boldsymbol{\phi}}$ is the probabilistic encoder, mapping each dualized constraint c in ι to a latent vector $\boldsymbol{z}_c$ computed by independent Gaussian distributions, $f_{\boldsymbol{\theta}}$ is the decoder map-

[1] Code in JULIA at https://github.com/FDemelas/Learning_Lagrangian_Multipliers.jl

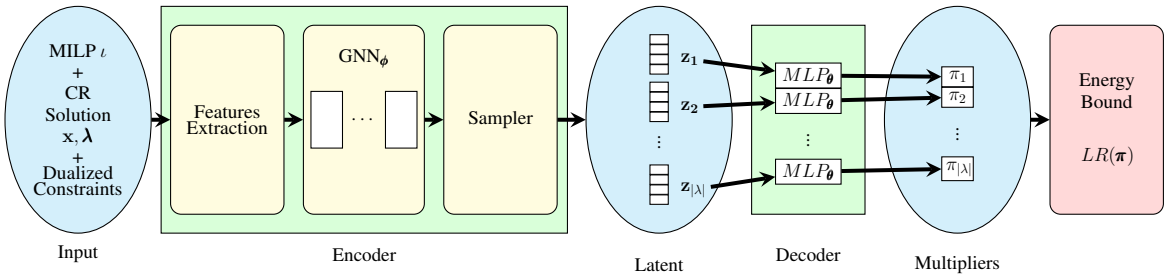

Figure 1. Overall Architecture. From the bipartite graph representation of a MILP and its CR solution, the model computes a Lagrangian dual solution. First the MILP is encoded by a GNN, from which we parameterize a sampler for constraint representations. These representations are then passed through a decoder to compute Lagrangian Multipliers.

ping each[2] $\boldsymbol{z}_c$ to its corresponding LM deviation δ_c from the CR dual value λ_c, and $[\cdot]_+$ is the component-wise softplus function. We can observe that this objective has the following properties amenable to gradient-based learning:

1. $LR(\boldsymbol{\pi})$ is bounded from above: optimal LMs $\boldsymbol{\pi}^*$ maximize $LR(\boldsymbol{\pi})$ over all possible LMs, that is $LR(\boldsymbol{\pi}^*) \geq LR(\boldsymbol{\pi})$ for any $\boldsymbol{\pi} = \boldsymbol{\lambda} + f_{\boldsymbol{\theta}}(\boldsymbol{z})$. Moreover, $LR(\boldsymbol{\pi})$ is a concave piece-wise linear function, in other words all optimal solutions will give the same bound.

2. It is straightforward to compute a subgradient w.r.t. to parameters $\boldsymbol{\theta}$: $\nabla_{\boldsymbol{\theta}} LR([\boldsymbol{\lambda} + f_{\boldsymbol{\theta}}(\boldsymbol{z})]_+; \iota)$ is equal to:

$$\left(\frac{\partial [\boldsymbol{\lambda} + f_{\boldsymbol{\theta}}(\boldsymbol{z})]_+}{\partial \boldsymbol{\theta}}\right)^\top \nabla_{\boldsymbol{\pi}} LR(\boldsymbol{\pi}; \iota)$$

   The Jacobian on the left is computed via backpropagation, while $LR(\boldsymbol{\pi}; \iota)$ is simple enough for a subgradient to be given analytically. Provided that $\bar{\boldsymbol{x}}$ is an optimal solution of the relaxed Lagrangian problem of ι associated with $\boldsymbol{\pi}$, we derive:

$$\nabla_{\boldsymbol{\pi}} LR(\boldsymbol{\pi}; \iota) = \boldsymbol{b} - \boldsymbol{A}\bar{\boldsymbol{x}}$$

   This means that in order to compute a subgradient for $\boldsymbol{\theta}$, we first need to solve each subproblem. Since subproblems are independent, this can be done in parallel.

3. For parameters $\boldsymbol{\phi}$, we again leverage function composition and the fact that $q_{\boldsymbol{\phi}}$ is a Gaussian distribution, so we can approximate the expectation by sampling and use the reparameterization trick (Kingma & Welling, 2014; Schulman et al., 2015) to perform standard backpropagation. We implement $q_{\boldsymbol{\phi}}$ as a neural network, described in details in the following section, returning a mean vector and a variance vector for each dualized constraint c, from which a sampler returns a representation vector $\boldsymbol{z}_c$. For numerical stability, the variance is clipped to a safe interval (Rybkin et al., 2021).

[2]With a slight abuse of notation, we use function $f : \mathbb{R}^m \to \mathbb{R}^n$ on *batches* of size p to become $\mathbb{R}^{m \times p} \to \mathbb{R}^{n \times p}$.

2.3. Encoding and Decoding Instances

Encoder One of the challenges in Machine Learning applications to Combinatorial Optimization is that instances have different input sizes, and so the encoder must be able to cope with these variations to produce high-quality features. Of course this is also the case in many other applications, for instance NLP where texts may differ in size, but there is no general consensus as to what a good feature extractor for MILP instances looks like, contrarily to other domains where variants of RNNs or Transformers have become the de facto standard encoders.

We depart from previous approaches to Lagrangian prediction (Sugishita et al., 2024) restricted to instances of the same size, and follow more generic approaches to MILP encoding such as (Gasse et al., 2019; Nair et al., 2020; Khalil et al., 2017) where each instance is converted into a bipartite graph and further encoded by GNNs to compute meaningful feature vectors associated with dualized constraints. Each MILP is converted to a bipartite graph composed of one node for each variable and one node for each constraint. There is an edge between a variable node n_v and a constraint node n_c if and only if v appears in c. Each node (variable or constraint) is represented by an initial feature vector $\boldsymbol{e}_n$. We use features similar to ones given in (Gasse et al., 2019).[3] Following Nair et al. (2020), variables and constraints are encoded as the concatenation of variable features followed by constraint features, of which only one is non-zero, depending on the type of nodes.

To design our stack of GNNs, we take inspiration from structured prediction models for images and texts, where Transformers (Vaswani et al., 2017) are ubiquitous. However, since our input has a bipartite graph structure, we replace the multihead self-attention layers with simple linear graph convolutions[4] (Kipf & Welling, 2017). Closer to our work, we follow Nair et al. (2020) which showed

[3]See Appendix A for more details.

[4]Alternatively, this can be seen as a masked attention, where the mask is derived from the input graph adjacency matrix.

that residual connections (He et al., 2016), dropout (Srivastava et al., 2014) and layer normalization (Ba et al., 2016) are important for the successful implementation of feature extractors for MILP bipartite graphs.

Before the actual GNNs, initial feature vectors $\{\boldsymbol{e}_n\}_n$ are passed through a MLP F to find feature combinations and extend node representations to high-dimensional spaces: $\boldsymbol{h}_n^0 = F(\boldsymbol{e}_n), \forall n$. Then interactions between nodes are taken into account by passing vectors through blocks, represented in Figure 2, consisting of two sublayers.

- The first sublayer connects its input via a residual connection to a layer normalization LN followed by a linear graph convolution $CONV$ of length 1, followed by a dropout regularization DO:

  $$\boldsymbol{h}'_n = \boldsymbol{h}_n + DO(CONV(LN(\boldsymbol{h}_n)))$$

  The graph convolution passes messages between nodes. In our context, it passes information from variables to constraints, and conversely.

- The second sublayer takes as input the result of first one, and connects it with a residual connection to a sequence made of a layer normalization LN, a MLP transformation and a dropout regularization DO:

  $$\boldsymbol{h}_n = \boldsymbol{h}'_n + DO(MLP(LN(\boldsymbol{h}'_n)))$$

  The MLP is in charge of finding non-linear interactions in the information collected in the previous sublayer.

This block structure, depicted in Figure 2, is repeated several times, typically 5 times in our experiments, in order to extend the domain of locality. The learnable parameters of a block are the parameters of the convolution in the first sublayer and the parameters of the MLP in the second one. Remark that we start each sublayer with normalization, as it has become the standard approach in Transformer recently (Chen et al., 2018). We note in passing that this has also been experimented with by Gasse et al. (2019) in the context of MILP, although only once before the GNN input, whereas we normalize twice per block, at each block.

Finally, the GNN returns the vectors associated with dualized constraints $\{\boldsymbol{h}_c\}_c$. Each vector $\boldsymbol{h}_c$ is interpreted as the concatenation of two vectors $[\boldsymbol{z}_\mu; \boldsymbol{z}_\sigma]$ from which we compute $\boldsymbol{z}_c = \boldsymbol{z}_\mu + \exp(\boldsymbol{z}_\sigma) \cdot \boldsymbol{\epsilon}$ where elements of $\boldsymbol{\epsilon}$ are sampled from the normal distribution. This concludes the implementation of the probabilistic encoder $q_{\boldsymbol{\phi}}$.

Decoder Recall that, in our architecture, from each latent vector representation $\boldsymbol{z}_c$ of dualized constraint c we want to compute the scalar deviation δ_c to the CR dual value λ_c so that the sum of the two improves the Lagrangian bound given by the CR dual solution. In other words, we want to compute $\boldsymbol{\delta}$ such as $\boldsymbol{\pi} = [\boldsymbol{\lambda} + \boldsymbol{\delta}]_+$ gives a *good* Lagrangian bound $LR(\boldsymbol{\pi})$. Its exact computation is of combinatorial nature and problem specific.[5]

The probabilistic nature of the encoder-decoder can be exploited further: during evaluation, when computing a Lagrangian Relaxation, we sample constraint representations 5 times from the probabilistic encoder and return the best $LR(\boldsymbol{\pi})$ value from the decoder.

Link with Energy Based Models in Structured Prediction The relaxed Lagrangian problem usually decomposes into independent subproblems due to the dualization of the linking constraints. In this case, for each independent Lagrangian subproblem we want to find its optimal variable assignment, usually with local combinatory constraints, for its objective reparameterized with $\boldsymbol{\pi}$. This approach is typical of structured prediction: we leverage neural networks to extract features in order to compute local energies (scalars), which are used by a combinatorial algorithm outputting a structure whose objective value can be interpreted as a global energy. For instance, this is reminiscent of how graph-based syntactic parsing models in NLP compute parse scores (global energies) as sums of arc scores (local energies) computed by RNNs followed by MLPs, where the choice of arcs is guided by well-formedness constraints enforced by a maximum spanning tree solver, see for instance (Kiperwasser & Goldberg, 2016). Thus, the decoder is local to each dualized constraint, and we leverage subproblems to interconnect predictions:

1. We compute LMs (local energies) $\pi_c = [\lambda_c + f_{\boldsymbol{\theta}}(\boldsymbol{z}_c)]_+$ for all dualized constraints c, where $f_{\boldsymbol{\theta}}$ is implemented as a feed-forward network computing the deviation.

2. For parameter learning or if the subproblem solutions or the Lagrangian bound are the desired output, vector $\boldsymbol{\pi}$ is then passed to the Lagrangian subproblems which compute independently and in parallel their local solutions $\boldsymbol{x}$ and the corresponding values are summed to give (global energy) $LR(\boldsymbol{\pi})$.

3. Related Work

There is growing interest in leveraging Machine Learning (ML) alongside optimization algorithms (Bengio et al., 2021), in particular with the goal of improving MILP solvers's efficiency (Zhang et al., 2023). Indeed, even though MILP solvers solve problems in an exact way, they make many heuristic decisions which can be based on data-driven ML systems. For instance, classifiers have been

[5] $LR(\boldsymbol{\pi})$ is described in Appendices B and C for the two problems on which we evaluate our method.

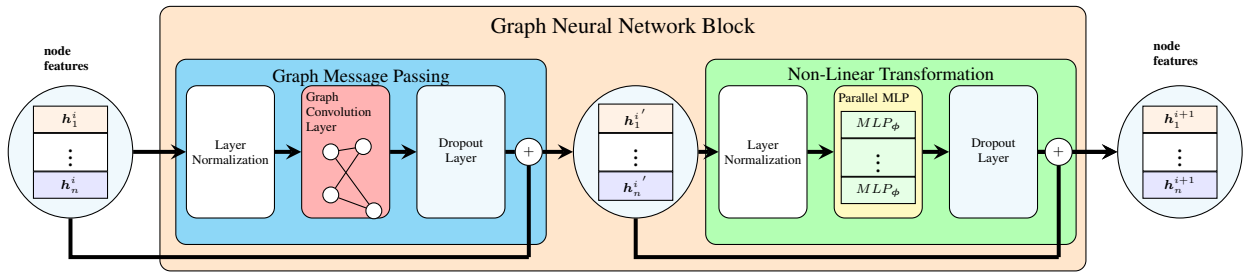

Figure 2. The Graph Neural Network block. The first part is graph message-passing: we apply layer normalization to node features, then convolution over the instance's bipartite graph representation and finally dropout. The second phase consists of normalization, a Multi-Layer perceptron in parallel over all the nodes of the bipartite graph, then dropout. Both sublayers use residual connection between input and output. We apply this block several times to improve feature representations.

designed for Branch and Bound (B&B) algorithms (Lodi & Zarpellon, 2017) in order to choose which variables to branch on (Alvarez et al., 2017; Khalil et al., 2016; He et al., 2014; Etheve et al., 2020), which B&B node to process (Yilmaz & Yorke-Smith, 2021; Labassi et al., 2022), to decide when to perform heuristics (Hottung et al., 2020; Khalil et al., 2017) or how to schedule them (Chmiela et al., 2021).

In this work, we depart from this main trend and predict a dual bound for MILP instances sharing common features, which can in turn be used to improve solvers. Several propositions have tackled the prediction of high quality primal and dual bounds. For instance, Nair et al. (2020) predict partial variable assignments, resulting in small MILPs which can be solved to optimality. Another way to provide primal solutions is to transform a MILP into an easier one, solve it and apply a procedure to recover primal feasibility (Dalle et al., 2022; Parmentier, 2022). Many works use Reinforcement Learning and guided greedy decoding to find high-quality approximate solutions for NP-hard problems, *e.g.* (Kool et al., 2019). For dual bounds, ML has been employed for cut selection in cutting planes algorithms (Baltean-Lugojan et al., 2018; Wang et al., 2023; Balcan et al., 2021; Berthold et al., 2022; Tang et al., 2020; Huang et al., 2022; Afia & Kabbaj, 2017, Tetouan Morocco; Morabit et al., 2021), an essential feature of MILP solvers which must balance strengthened linear relaxations with increased computations due to added cuts (Dey & Molinaro, 2018).

Regarding specifically prediction for Lagrangian dual solutions, Nair et al. (2018) consider 2-stage stochastic MILPs, approached by a Lagrangian decomposition for which they learn to predict LMs compliant with any second-stage scenario to give a good bound on average. Lange & Swoboda (2021) propose a heuristic to solve binary ILPs based on a specific LD where the relaxed LR problem is decomposed into many subproblems, one per constraint, solved using binary decision diagrams. This method is modified by Abbas & Swoboda (2022) to be run on GPU. The block coordinate method used to heuristically solve LD is improved by learning parameters used for initializing and updating Lagrangian multipliers (Abbas & Swoboda, 2024). In contrast to our generic method, other previous attempts at Lagrangian dual solution prediction for deterministic MILPs focus on a specific combinatorial optimization problem, such as the cutting stock problem (Kraul et al., 2023), where a MLP predicts the dual Lagrangian value for each constraint (*i.e.* stock) separately, or the unit commitment problem (Sugishita et al., 2024), where the same problem is solved daily but with different demand forecasts with either a MLP or a random forest which predicts dual solutions used to warm-start BM.

In our work, we assume that the set of dualized constraints is given, but predicting such a set is also an active avenue of research where solutions must find a good compromise between the quality of the Lagrangian dual bound and the running time to compute this bound (Kruber et al., 2017; Basso et al., 2020).

Regarding our use of GNNs, this has become a common MILP feature extractor in recent works, either on the factor bipartite graph (Gasse et al., 2019; Nair et al., 2020) or directly on the underlying graph in routing problems (Sun & Yang, 2023). While these works use GNNs to extract features for variable predictions, we use GNNs to extract features for constraints, which are then decoded to Lagrangian Multipliers. Our specific GNN architecture is based on the block structure of Transformers (Vaswani et al., 2017) where attention is replaced by a linear graph convolution.

Our method predicts a deviation from an initial solution, and can also be understood as predicting a gradient or subgradient step. We can thus relate our approach to works on gradient descent (Andrychowicz et al., 2016; Ba et al., 2022) and unrolling of iterative methods for structured prediction (Yang et al., 2016; Belanger et al., 2017). This is also related to amortization especially in relation with subgradient methods already studied in the ML community (Komodakis et al., 2014; Meshi et al., 2010). However, in previous works amortization was performed only during training, and iterative methods were used at testing time.

In the case of semi-amortization, our method can be used as an informed starting point for a descent algorithm applied to quadratic optimization (Sambharya et al., 2022), or probabilistic inference (Kim et al., 2018).

4. Evaluation

We evaluate our approach[6] on two standard problems of Operations Research, namely Multi-Commodity Fixed-Charge Network Design and Generalized Assignment.

4.1. Problems and Datasets

We review briefly the two problems and the data generation process. More details on MILP formulations and Lagrangian relaxations can be found in Appendices B and C and a thorough description of dataset generation is given in Appendix D.

Multi-Commodity Fixed-Charge Network Design (MC) Given a network with arc capacities and a set of commodities, MC consists in activating a subset of arcs and routing each commodity from its origin to its destination, possibly fractioned on several paths, using only the activated arcs. The objective is to minimize the total cost induced by the activation of arcs and the routing of commodities. This problem has been used in many real-world applications for a long time, see for instance (Magnanti & Wong, 1984) for telecommunications. It is NP-hard and its continuous relaxation provides poor bounds when arc capacities are high. Hence, it is usually tackled with Lagrangian relaxation-based methods (Akhavan Kazemzadeh et al., 2022).

While the Canad dataset is the standard and well-established dataset of instances for evaluating MC solvers (Crainic et al., 2001), it is too small to be used as a training set for Machine Learning where large collections of instances sharing common features are required. Thus, we generate new instances from a subset of instances of the Canad dataset (Crainic et al., 2001), that we divide into four datasets of increasing difficulty. The first two datasets, MC-SML-40 and MC-SML-VAR, contain instances that all share the same network (20 nodes and 230 edges) and the same arc capacities and fixed costs, but with different values for origins, destinations, volumes, and routing costs. Instances of the former all involve the same number of commodities (40), while for the latter the number of commodities varies from 40 to 200. Dataset MC-BIG-40 is generated similarly to MC-SML-40 but upon a bigger graph containing 30 nodes and 520 arcs. Finally, MC-BIG-VAR contains examples generated using either the network of MC-SML-40 or the one of MC-BIG-40, with the number of commodities varying between 40 and 200.

[6] See Appendix E for hyperparameter values used in our experiments.

Generalized Assignment (GA) GA consists, given a set of items and a set of capacitated bins, in assigning items to bins without exceeding their capacity in order to maximize the profit of the assignment. GA is a well-known problem in Operations Research and has numerous applications such as job-scheduling in Computer Science (Balachandran, 1976), distributed caching (Fleischer et al., 2011) or even parking allocation (Mladenović et al., 2020).

We generated two datasets, namely GA-10-100 and GA-20-400, containing respectively instances with 10 bins and 100 items, and with 20 bins and 400 items. Weights, profits and bin capacities are sampled using a distribution determined from values of standard instances (Yagiura et al., 1999).

4.2. Numerical Results

We want to evaluate how our Lagrangian bound prediction compares to an iterative model based on subgradient, and how useful the former is as an initial point to warm-start the latter. For that purpose, we choose a state-of-the-art proximal bundle solver provided by SMS++ (Frangioni et al., 2023) which allows writing a MILP in a block structure fashion and using decomposition techniques to solve subproblems efficiently. We also compare our approach with CR computed using the CPLEX[7] optimizer.

All MILP instances for which we want to evaluate our model are first solved by SMS++. For an instance ι we denote $\boldsymbol{\pi}^*_\iota$ the LMs returned by SMS++.

Metrics We use the percentage gap as metrics to evaluate the quality of the bounds computed by the different systems, averaged over a dataset of instances $\mathcal{I}$. For a system returning a bound B_ι for an instance ι the percentage GAP is:

$$100 \times \frac{1}{|\mathcal{I}|} \sum_{\iota \in \mathcal{I}} \frac{LR(\boldsymbol{\pi}^*_\iota) - B_\iota}{LR(\boldsymbol{\pi}^*_\iota)}$$

GAP measures the quality of the bound B_ι, and is zero when B_ι equals the optimal Lagrangian bound.

Data for Evaluation We divide each dataset of 2000 instances in train (80%), validation (10%) and test (10%). Parameters are learned on the train set, model selection is performed on validation set, and test proxies for unseen data. Results are averaged over 3 random initializations.

Bound Accuracy Table 1 reports the performance of different systems on our 6 datasets. We compare the bound returned by CR, and the bound of the Lagrangian relaxed

[7] https://www.ibm.com/products/ilog-cplex-optimization-studio

problem obtained with LMs computed by different methods:

- LR(0) is the LR value computed with LMs set to zero.
- LR(CR) is the LR value computed with LMs set to CR dual solution.
- LR(k-NN) is the LR value computed with LMs set to the average value of LMs from the train set returned by a k-NN regressor.[8]
- LR(MLP) is the LR value computed with LMs returned by a MLP[9] instead of the GNN-based encoder-decoder.
- Ours, that is the LR value computed with LMs set the output of our encoder-decoder. As written in the previous section we sample 5 LM assignments per instance and return the best LR value.

For all datasets, our method outperforms other approaches. Our model can reach 2% difference with BM on MC-SML-40, the easiest corpus with a small fixed network and a fixed number of commodities. This means that one pass through our network can save numerous iterations if we can accept about 2% bound error on average. The margin with other methods is quite large for MC datasets where the CR bound is far from the optimum, with a gap reduction ranging from 77% (MC-BIG-VAR) to 84% (MC-SML-40) depending on the dataset. For GA, where CR is closer to the optimum, our model still manages to find better solutions. Even though the gap absolute difference may seem small, the gap reduction from the second-best model LR(CR) ranges from 30% (GA-10-100) to 44% (GA-20-400), a significant error reduction.

Compared to simpler ML approaches, we see *(i)* that retrieving LM values from k-NN clustering is not a viable solution, even when the validation instances are close to the training instances (MC-SML-40), clustering cannot find meaningful neighbors, and *(ii)* the graph feature extractor (GNN) is paramount: the LR(MLP) architecture seems unable to deviate LMs consistently from CR solutions and can even perform worse than CR or LR(CR) (GA-20-400).

Regarding speed, LR(0) is the fastest since it simply amounts to solving the relaxed Lagrangian problem with the original costs. Then CR and LR(CR) are second, the difference being that for the latter after solving CR, the dual solution $\boldsymbol{\lambda}$ is used to compute the $LR(\boldsymbol{\lambda})$. Slowness for LR(MLP) and LR(k-NN) is mainly caused by feature extraction (*cf*. Appendix G).

[8] See Appendices F and G for more information about the implemented k-NN method.

[9] Additional initial features that the ones used in our model are used, see Appendix G for more details.

Table 1. Bound accuracies of different methods on test sets averaged by instance.

Dataset	Methods	GAP %	time (ms)
MC-SML-40	CR	12.99	90.63
	LR(0)	100.00	**0.35**
	LR(CR)	12.97	90.98
	LR(k-NN)	38.80	219.42
	LR(MLP)	10.70	142.48
	ours	**2.09**	120.96
MC-SML-VAR	CR	22.29	283.63
	LR(0)	100.00	**1.32**
	LR(CR)	22.29	285.03
	LR(k-NN)	44.12	371.51
	LR(MLP)	16.71	369.61
	ours	**4.42**	374.20
MC-BIG-40	CR	15.94	220.91
	LR(0)	100.00	**0.75**
	LR(CR)	15.85	229.57
	LR(k-NN)	54.57	334.99
	LR(MLP)	13.67	556.89
	ours	**4.20**	283.40
MC-BIG-VAR	CR	20.66	287.20
	LR(0)	100.00	**1.37**
	LR(CR)	20.63	288.55
	LR(k-NN)	49.74	886.91
	LR(MLP)	16.14	515.60
	ours	**4.77**	374.78
GA-10-100	CR	1.91	9.59
	LR(0)	3.13	**0.44**
	LR(CR)	0.79	10.15
	LR(k-NN)	1.07	11.70
	LR(MLP)	0.78	51.71
	ours	**0.55**	16.19
GA-20-400	CR	0.44	71.40
	LR(0)	2.70	**7.51**
	LR(CR)	0.27	78.80
	LR(k-NN)	0.43	89.68
	LR(MLP)	0.28	114.41
	ours	**0.15**	124.96

Warm-starting Iterative Solvers We want to test whether the Lagrangian Multipliers predicted by our model can be used as an informed starting point for an iterative solver for the Lagrangian Dual LD, namely the bundle method as implemented by SMS++ and the subgradient method. While the latter is simple to implement and only requires solving $LR(\boldsymbol{\pi})$, it has a non-smooth objective and the subgradient does not always give a descent direction, resulting in unstable updates. In contrast, the bundle method is stabilized with a quadratic penalty assuring a smooth objective, at the expense of longer computation times. We hope that our model can produce good starting points for both methods and thus avoid many early iterations.

In Table 2 we compare different initial LM vectors on the validation set of MC-BIG-VAR for the bundle method. We run our bundle solver until the difference between $LR(\boldsymbol{\pi}^*)$ and the current bound is smaller than the threshold ϵ. We average resolution times and numbers of iterations over instances, and compute standard deviation. We compare three initialization methods: zero, using CR dual solutions, and our model's predictions.

Table 2. Impact of initialization for a Bundle solver on MC-BIG-VAR. We consider initializations from the null vector (zero), the continuous relaxation duals (CR), and our model (Ours).

ϵ	zero		CR		Ours	
	time (s)	# iter.	time (s)	# iter.	time (s)	# iter.
1e-1	34.34 (±81.22)	90.12 (±52.41)	31.67 (±75.12)	83.00 (±50.73)	**16.09** (± 42.79)	**60.39** (± 41.37)
1e-2	68.80 (±188.21)	141.43 (±112.72)	62.09 (±171.71)	133.26 (±109.60)	**36.10** (± 106.22)	**105.93**(± 97.04)
1e-3	100.71 (±288.16)	188.14 (±167.33)	89.26 (±251.15)	179.40 (±170.15)	**57.23** (± 177.24)	**143.36** (± 142.58)
1e-4	105.03 (±298.53)	207.90 (±198.92)	101.14 (±283.52)	200.42 (±197.60)	**63.25** (± 190.47)	**159.42** (± 162.32)

We can see that CR is not competitive with the null initialization, since the small gain in the number of iterations is absorbed by the supplementary computation. However, our model's predictions give a significant improvement over the other two initialization methods, despite the additional prediction time. Resolution time is roughly halved for the coarsest threshold, and above one-third faster for the finest one. This is expected, as gradient-based methods naturally slow down as they approach convergence. In appendix I we perform the same experiments as in Table 2 for the subgradient method.

Ablation Study In Table 3 we compare three variants of our original model, denoted `ours`, on MC-SML-40 and MC-BIG-VAR. Results are averaged over 3 runs.

In the first variant `-max`, instead of sampling multiple LMs for each dualized constraint and keeping the best, we take one sample only per constraint. We can see that this has a minor incidence on the quality of the returned solution. In `-sum`, the dual solution values are passed as constraint node features but are not added to the output of the decoder to produce LMs, *i.e.* the network must transport these values from its input layer to its output. This has a sensible negative impact of GAP scores. In the third variant, `-cr` the CR solution is not given as input features to the network (nor added to the network's output). This is challenging because the network does not have access to a good starting point, this is equivalent to initializing LMs to zero. The last variant, `-sample`, uses CR as `ours` but does not sample representations $\boldsymbol{z}_c$ in the latent domain. We interpret the vector $\boldsymbol{h}_c$ associated with dualized constraint c after the GNN stack directly as vector $\boldsymbol{z}_c$, making the encoder deterministic.

We can see that the performance of `-sum` just below `ours`, while `-cr` cannot return competitive bounds. This indicates that the CR solution passed as input features is essential for our architecture to get good performance, whereas the computation of the deviation instead of the full LM directly is not an important trait. Still, we note that performances of `-cr` should be compared with $LR(0)$ in Table 1 rather than $LR(CR)$. In that case, we see that the GAP reduction is around 80%, making it clear that our model is not simply repeating CR solutions. This means that our model could be used without the CR solution information as input, opening our methods to a wider range of problems, and paving the way for faster models.

Table 3. Ablation studies comparing the prediction of our model with, predicting LMs on rather than deviation from CR (`-sum`), not using CR features at all (`-cr`), or replacing the probabilistic encoder by a deterministic one (`-sample`).

	GAP %	
model	MC-SML-40	MC-BIG-VAR
`ours`	2.09	4.77
`-max`	2.10	4.79
`-sum`	2.63	6.77
`-cr`	20.26	23.78
`-sample`	2.18	5.86

Table 4. Generalization results over bigger instances.

	GAP %		time (s)	
# commodities	Ours	LR(CR)	Ours	LR(CR)
160	6.51	27.85	1.533	0.8915
200	7.62	30.18	1.4328	1.0889

Finally, `-sample` is a system trained without sampling at training time, *i.e.* the encoder-decoder is deterministic. We see that sampling gives a slight performance increase on small and bigger instances.

In Appendix H we compare the architecture we introduce in this work with the architecture proposed by Nair et al. in (Nair et al., 2018) and the one presented in (Gasse et al., 2019), for several layers from 1 to 10.

Generalization Properties We test the model trained on MC-BIG-VAR on a dataset composed of 1000 bigger instances. They are created using the biggest graph used to generate the MC-BIG-VAR dataset but contain 160 or 200 commodities whereas the instances of MC-BIG-VAR with the same graphs only contain up to 120 commodities. In Table 4 we can see that our model still performs well in these instances dividing by 4 the gap provided by LR(CR).

5. Conclusion

We have presented a novel method to compute good Lagrangian dual solutions for MILPs sharing common attributes, by predicting Lagrangian multipliers. We cast this problem as an encoder-decoder prediction, where the probabilistic encoder outputs one distribution per dualized constraint from which we sample constraint vector representation. Then a decoder transforms these representations into Lagrangian multipliers.

We experimentally showed that this method gives bounds significantly better than the commonly used heuristics on two standard combinatorial problems: it reduces the continuous relaxation gap to the optimal bound up to 85%, and when used to warm-start an iterative solver, the points predicted by our models reduce solving times by a large margin.

Our predictions could be exploited in primal heuristics, possibly with auxiliary losses predicting values from variable nodes, or to efficiently guide a Branch-and-Bound exact search. Predictions could be stacked to act as an unrolled iterative solver. Finally, we can see our model as performing denoising from a previous solution and could be adapted to fit in a diffusion model.

Acknowledgments

The authors acknowledge the support of the French Agence Nationale de la Recherche (ANR), under grant ANR-23-CE23-0005 (project SEMIAMOR).

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Abbas, A. and Swoboda, P. Fastdog: Fast discrete optimization on gpu. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 439–449, June 2022.

Abbas, A. and Swoboda, P. Doge-train: Discrete optimization on GPU with end-to-end training. In Wooldridge, M. J., Dy, J. G., and Natarajan, S. (eds.), *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pp. 20623–20631. AAAI Press, 2024.

Afia, A. E. and Kabbaj, M. M. Supervised learning in branch-and-cut strategies. In *International Conference on Big Data Cloud and Applications*, 2017, Tetouan Morocco.

Akhavan Kazemzadeh, M. R., Bektaş, T., Crainic, T. G., Frangioni, A., Gendron, B., and Gorgone, E. Node-based lagrangian relaxations for multicommodity capacitated fixed-charge network design. *Discrete Applied Mathematics*, 308:pp. 255–275, 2022. Combinatorial Optimization ISCO 2018.

Alvarez, A. M., Louveaux, Q., and Wehenkel, L. A machine learning-based approximation of strong branching. *INFORMS J. Comput.*, 29(1):185–195, 2017.

Andrychowicz, M., Denil, M., Gómez, S., Hoffman, M. W., Pfau, D., Schaul, T., Shillingford, B., and de Freitas, N. Learning to learn by gradient descent by gradient descent. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

Ba, J., Erdogdu, M. A., Suzuki, T., Wang, Z., Wu, D., and Yang, G. High-dimensional asymptotics of feature learning: How one gradient step improves the representation. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 37932–37946. Curran Associates, Inc., 2022.

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization, 2016.

Balachandran, V. An Integer Generalized Transportation Model for Optimal Job Assignment in Computer Networks. *Operations Research*, 24(4):742–759, August 1976.

Balcan, M.-F. F., Prasad, S., Sandholm, T., and Vitercik, E. Sample complexity of tree search configuration: Cutting planes and beyond. *Advances in Neural Information Processing Systems*, 34:pp. 4015–4027, 2021.

Baltean-Lugojan, R., Bonami, P., Misener, R., and Tramontani, A. Selecting cutting planes for quadratic semidefinite outer-approximation via trained neural networks. In *optimization-online*, 2018.

Basso, S., Ceselli, A., and Tettamanzi, A. Random sampling and machine learning to understand good decompositions. *Annals of Operations Research*, 284(2):pp. 501–526, January 2020.

Beasley, J. E. A lagrangian heuristic for set-covering problems. *Naval Research Logistics (NRL)*, 37(1):pp. 151–164, 1990.

Belanger, D., Yang, B., and McCallum, A. End-to-end learning for structured prediction energy networks. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pp. 429–439. PMLR, 2017.

Bengio, Y., Lodi, A., and Prouvost, A. Machine learning for combinatorial optimization: a methodological tour d'horizon. *European Journal of Operational Research*, 290(2):pp. 405–421, 2021.

Berthold, T., Francobaldi, M., and Hendel, G. Learning to use local cuts. *arXiv preprint arXiv:2206.11618*, 2022.

Chen, M. X., Firat, O., Bapna, A., Johnson, M., Macherey, W., Foster, G., Jones, L., Schuster, M., Shazeer, N., Parmar, N., Vaswani, A., Uszkoreit, J., Kaiser, L., Chen, Z., Wu, Y., and Hughes, M. The best of both worlds: Combining recent advances in neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 76–86, Melbourne, Australia, July 2018. Association for Computational Linguistics.

Chmiela, A., Khalil, E. B., Gleixner, A., Lodi, A., and Pokutta, S. Learning to schedule heuristics in branch and bound. *Advances in Neural Information Processing Systems*, 34:pp. 24235–24246, 2021.

Conforti, M., Cornuéjols, G., and Zambelli, G. *Integer Programming*. Springer, New York, 2014.

Crainic, T. G., Frangioni, A., and Gendron, B. Bundle-based relaxation methods for multicommodity capacitated fixed charge network design. *Discrete Applied Mathematics*, 112(1-3):pp. 73–99, 2001.

Dalle, G., Baty, L., Bouvier, L., and Parmentier, A. Learning with combinatorial optimization layers: a probabilistic approach. *arXiv preprint arXiv:2207.13513*, 2022.

Dey, S. S. and Molinaro, M. Theoretical challenges towards cutting-plane selection. *Mathematical Programming*, 170 (1):pp. 237–266, July 2018. ISSN 0025-5610, 1436-4646.

Etheve, M., Alès, Z., Bissuel, C., Juan, O., and Kedad-Sidhoum, S. Reinforcement learning for variable selection in a branch and bound algorithm. In Hebrard, E. and Musliu, N. (eds.), *Integration of Constraint Programming, Artificial Intelligence, and Operations Research - 17th International Conference, CPAIOR 2020, Vienna, Austria, September 21-24, 2020, Proceedings*, volume 12296 of *Lecture Notes in Computer Science*, pp. 176–185. Springer, 2020.

Fleischer, L., Goemans, M. X., Mirrokni, V. S., and Sviridenko, M. Tight approximation algorithms for maximum separable assignment problems. *Mathematics of Operations Research*, 36(3):pp. 416–431, 2011. ISSN 0364765X, 15265471.

Frangioni, A., Iardella, N., and Durbano Lobato, R. SMS++, 2023. URL https://gitlab.com/smspp/smspp-project.

Gasse, M., Chételat, D., Ferroni, N., Charlin, L., and Lodi, A. Exact Combinatorial Optimization with Graph Convolutional Neural Networks. In Wallach, H., Larochelle, H., Beygelzimer, A., Alché-Buc, F. d., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Gendron, B., Crainic, T. G., and Frangioni, A. *Multicommodity Capacitated Network Design*, pp. 1–19. Springer US, Boston, MA, 1999.

Geoffrion, A. M. Lagrangean relaxation for integer programming. In Balinski, M. L. (ed.), *Approaches to Integer Programming*, pp. 82–114. Springer Berlin Heidelberg, Berlin, Heidelberg, 1974. ISBN 978-3-642-00740-8.

Golden, B., Raghavan, S., and Wasil, E. (eds.). *The Vehicle Routing Problem: Latest Advances and New Challenges*, volume 43 of *Operations Research/Computer Science Interfaces*. Springer US, Boston, MA, 2008. ISBN 978-0-387-77777-1 978-0-387-77778-8.

He, H., Daume III, H., and Eisner, J. M. Learning to search in branch and bound algorithms. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and Weinberger, K. (eds.), *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

He, K., Zhang, X., Ren, S., and Sun, J. Deep Residual Learning for Image Recognition. In *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '16, pp. 770–778. IEEE, June 2016.

Hiriart-Urruty, J.-B. and Lemaréchal, C. *Convex analysis and minimization algorithms II: Advance Theory and Bundle Methods*, volume 305. Springer science & business media, 1996.

Hottung, A., Tanaka, S., and Tierney, K. Deep learning assisted heuristic tree search for the container premarshalling problem. *Computers & Operations Research*, 113:104781, 2020.

Huang, Z., Wang, K., Liu, F., Zhen, H.-L., Zhang, W., Yuan, M., Hao, J., Yu, Y., and Wang, J. Learning to select cuts for efficient mixed-integer programming. *Pattern Recognition*, 123:108353, 2022. ISSN 0031-3203.

Khalil, E., Le Bodic, P., Song, L., Nemhauser, G., and Dilkina, B. Learning to branch in mixed integer programming. *Proceedings of the AAAI Conference on Artificial Intelligence*, 30(1), Feb. 2016.

Khalil, E. B., Dilkina, B., Nemhauser, G. L., Ahmed, S., and Shao, Y. Learning to run heuristics in tree search. In *Ijcai*, pp. 659–666, 2017.

Kim, Y., Wiseman, S., Miller, A., Sontag, D., and Rush, A. Semi-amortized variational autoencoders. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2678–2687. PMLR, 10–15 Jul 2018.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. In Bengio, Y. and LeCun, Y. (eds.), *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

Kiperwasser, E. and Goldberg, Y. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics*, 4:pp. 313–327, 2016.

Kipf, T. and Welling, M. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

Komodakis, N., Xiang, B., and Paragios, N. A Framework for Efficient Structured Max-Margin Learning of High-Order MRF Models. Technical Report 7, 2014.

Kool, W., van Hoof, H., and Welling, M. Attention, learn to solve routing problems! In *International Conference on Learning Representations*, 2019.

Korte, B. H. and Vygen, J. *Combinatorial Optimization: Theory and Algorithms*. Number v. 21 in Algorithms and Combinatorics. Springer, Heidelberg ; New York, 5th ed edition, 2012.

Kraul, S., Seizinger, M., and Brunner, J. O. Machine learning–supported prediction of dual variables for the cutting stock problem with an application in stabilized column generation. *INFORMS Journal on Computing*, 35(3):pp. 692–709, 2023.

Kruber, M., Lübbecke, M. E., and Parmentier, A. Learning when to use a decomposition. In Salvagnin, D. and Lombardi, M. (eds.), *Integration of AI and OR Techniques in Constraint Programming - 14th International Conference, CPAIOR 2017, Padua, Italy, June 5-8, 2017, Proceedings*, volume 10335 of *Lecture Notes in Computer Science*, pp. 202–210. Springer, 2017.

Labassi, A. G., Chételat, D., and Lodi, A. Learning to compare nodes in branch and bound with graph neural networks. *Advances in Neural Information Processing Systems*, 35:pp. 32000–32010, 2022.

Lange, J.-H. and Swoboda, P. Efficient message passing for 0–1 ilps with binary decision diagrams. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. pp. 6000–6010. PMLR, 18–24 Jul 2021.

Le, Q., Smola, A., and Vishwanathan, S. Bundle methods for machine learning. *Advances in neural information processing systems*, 20, 2007.

Le Cun, Y., Chopra, S., Hadsell, R., Ranzato, M., and Huang, F. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.

Lodi, A. and Zarpellon, G. On learning and branching: a survey. *Top*, 25:pp. 207–236, 2017.

Magnanti, T. L. and Wong, R. T. Network design and transportation planning: Models and algorithms. *Transp. Sci.*, 18(1):pp. 1–55, 1984.

Meshi, O., Sontag, D., Jaakkola, T., and Globerson, A. Learning efficiently with approximate inference via dual losses. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pp. 783–790, Madison, WI, USA, 2010. Omnipress. ISBN 9781605589077.

Mladenović, M., Delot, T., Laporte, G., and Wilbaut, C. The parking allocation problem for connected vehicles. *Journal of Heuristics*, 26(3):377–399, June 2020. ISSN 1381-1231, 1572-9397.

Morabit, M., Desaulniers, G., and Lodi, A. Machine-Learning–Based Column Selection for Column Generation. *Transportation Science*, 55(4):815–831, July 2021. ISSN 0041-1655, 1526-5447.

Nair, V., Dvijotham, D., Dunning, I., and Vinyals, O. Learning fast optimizers for contextual stochastic integer programs. In *UAI*, pp. 591–600, 2018.

Nair, V., Bartunov, S., Gimeno, F., von Glehn, I., Lichocki, P., Lobov, I., O'Donoghue, B., Sonnerat, N., Tjandraatmadja, C., Wang, P., Addanki, R., Hapuarachchi, T., Keck, T., Keeling, J., Kohli, P., Ktena, I., Li, Y., Vinyals, O., and Zwols, Y. Solving mixed integer programs using neural networks. *CoRR*, abs/2012.13349, 2020.

Parmentier, A. Learning to approximate industrial problems by operations research classic problems. *Oper. Res.*, 70 (1):606–623, 2022.

Polyak, B. *Introduction to Optimization*. Optimization Software, New York, 1987.

Rybkin, O., Daniilidis, K., and Levine, S. Simple and effective vae training with calibrated decoders. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 9179–9189. PMLR, 18–24 Jul 2021.

Sambharya, R., Hall, G., Amos, B., and Stellato, B. End-to-end learning to warm-start for real-time quadratic optimization. In *Conference on Learning for Dynamics & Control*, 2022.

Schulman, J., Heess, N., Weber, T., and Abbeel, P. Gradient estimation using stochastic computation graphs. In Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):pp. 1929–1958, 2014.

Sugishita, N., Grothey, A., and McKinnon, K. Use of Machine Learning Models to Warmstart Column Generation for Unit Commitment. *INFORMS Journal on Computing*, January 2024. ISSN 1091-9856, 1526-5528.

Sun, Z. and Yang, Y. DIFUSCO: Graph-based diffusion solvers for combinatorial optimization. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Tang, Y., Agrawal, S., and Faenza, Y. Reinforcement learning for integer programming: Learning to cut. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 9367–9376. PMLR, 13–18 Jul 2020.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Wang, Z., Li, X., Wang, J., Kuang, Y., Yuan, M., Zeng, J., Zhang, Y., and Wu, F. Learning cut selection for mixed-integer linear programming via hierarchical sequence model. In *The Eleventh International Conference on Learning Representations*, 2023.

Wolsey, L. A. *Integer Programming*. Wiley, Hoboken, NJ, second edition edition, 2021.

Yagiura, M., Yamaguchi, T., and Ibaraki, T. *A Variable Depth Search Algorithm for the Generalized Assignment Problem*, pp. 459–471. Springer US, Boston, MA, 1999. ISBN 978-1-4615-5775-3.

Yang, Y., Sun, J., Li, H., and Xu, Z. Deep admm-net for compressive sensing mri. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

Yilmaz, K. and Yorke-Smith, N. A study of learning search approximation in mixed integer branch and bound: Node selection in scip. *AI*, 2(2):pp. 150–178, 2021. ISSN 2673-2688.

Zhang, J., Liu, C., Li, X., Zhen, H.-L., Yuan, M., Li, Y., and Yan, J. A survey for solving mixed integer programming via machine learning. *Neurocomputing*, 519:pp. 205–217, 2023.

A. Initial Features

To extract useful features, we define a network based on graph convolutions presented in Figure 1 in the line of the work of (Gasse et al., 2019) on MILP encoding. We detail the initial node features $\{\boldsymbol{e}_n\}_n$ of the MILP-encoding bipartite graph presented in Section 2.3.

Given an instance of the form:

$$(P) \qquad \min_{\boldsymbol{x}} \boldsymbol{w}^\top \boldsymbol{x} \tag{2a}$$

$$\boldsymbol{A}\boldsymbol{x} \begin{pmatrix} \geq \\ = \end{pmatrix} \boldsymbol{b} \tag{2b}$$

$$\boldsymbol{x} \in \mathbb{R}^m_+ \times \mathbb{N}^p \tag{2c}$$

we consider the following initial features for a variable x_j:

- its coefficient w_j in the objective function;
- its value in the primal solution of CR;
- its reduced cost $\bar{c}_j = w_j - \boldsymbol{\lambda}^T \boldsymbol{A}_j$ in CR where $\boldsymbol{A}_j$ is the j^{th} column of $\boldsymbol{A}$ and $\boldsymbol{\lambda}$ is the dual solution of CR;
- a binary value indicating whether x_j is integral or continuous.

For constraint $\boldsymbol{a}^\top \boldsymbol{x} \begin{pmatrix} \geq \\ = \end{pmatrix} b$ of (2b), we consider:

- the right-hand side b of the constraint;
- the value of the associated dual solution in CR;
- one binary value indicating whether the constraint is an equality or an inequality;
- one binary value stating whether c is dualized in the relaxed Lagrangian problem.

We use for each node n of the bipartite graph a feature vector $\boldsymbol{e}_n \in \mathbb{R}^8$. The first four components are used to encode the initial features if n corresponds to a variable and are set to 0 otherwise, whereas the next four components are used only if n is associated with a constraint and are set to 0 otherwise.

B. Multi Commodity Capacitated Network Design Problem

A MC instance is given by a directed simple graph $D = (N, A)$, a set of commodities K, an arc-capacity vector c, and two cost vectors r and f. Each commodity $k \in K$ corresponds to a triplet (o^k, d^k, q^k) where $o^k \in N$ and $d^k \in N$ are the nodes corresponding to the origin and the destination of commodity k, and $q^k \in \mathbb{N}^*$ is its volume. For each arc, $(i, j) \in A$, $c_{ij} > 0$ corresponds to the maximum amount of flow that can be routed through (i, j) and $f_{ij} > 0$ corresponds to the fixed cost of using arc (i, j) to route commodities. For each arc $(i, j) \in A$ and each commodity $k \in K$, $r^k_{ij} > 0$ corresponds to the cost of routing one unit of commodity k through arc (i, j).

A MC solution consists of an arc subset $A' \subseteq A$ and, for each commodity $k \in K$, in a flow of value q^k from its origin o^k to its destination d^k with the following requirements: all commodities are only routed through arcs of A', and the total amount of flow routed through each arc $(i, j) \in A'$ does not exceed its capacity c_{ij}. The solution cost is the sum of the fixed costs over the arcs of A' plus the routing cost, the latter being the sum over all arcs $(i, j) \in A$ and all commodities $k \in K$ of the unitary routing cost r^k_{ij} multiplied by the amount of flow of k routed through (i, j).

B.1. MILP formulation

A standard model for the MC problem (Gendron et al., 1999) introduces two sets of variables: the continuous flow variables x^k_{ij} representing the amount of commodity k that is routed through arc (i, j) and the binary design variables y_{ij} representing whether or not arc (i, j) is used to route commodities. Denoting respectively by $N^+_i = \{j \in N \mid (i, j) \in A\}$ and $N^-_i = \{j \in N \mid (j, i) \in A\}$ the sets of forward and backward neighbors of a vertex $i \in N$, the MC problem can be modeled as follows:

$$\min_{\boldsymbol{x},\boldsymbol{y}} \sum_{(i,j)\in A} \left(f_{ij} y_{ij} + \sum_{k \in K} r^k_{ij} x^k_{ij} \right) \tag{3a}$$

$$\sum_{j \in N^+_i} x^k_{ij} - \sum_{j \in N^-_i} x^k_{ji} = b^k_i \quad \forall i \in N, \forall k \in K \tag{3b}$$

$$\sum_{k \in K} x^k_{ij} \leq c_{ij} y_{ij}, \qquad \forall (i, j) \in A \tag{3c}$$

$$x^k_{ij} = 0 \qquad \forall k \in K, \forall (i, j) \in A \text{ s.t. } i = d^k \text{ or } j = o^k \tag{3d}$$

$$0 \leq x^k_{ij} \leq q^k \qquad \forall (i, j) \in A, \forall k \in K \tag{3e}$$

$$y_{ij} \in \{0, 1\}, \qquad \forall (i, j) \in A \tag{3f}$$

where

$$b^k_i = \begin{cases} q^k & \text{if } i = o^k, \\ -q^k & \text{if } i = d^k, \\ 0 & \text{otherwise.} \end{cases}$$

The objective function (3a) minimizes the sum of the routing and fixed costs. Equations (3b) are the flow conservation constraints that properly define the flow of each commodity through the graph. Constraints (3c) are the capacity constraints ensuring that the total amount of flow routed through each arc does not exceed its capacity or is zero if the arc is not used to route commodities. Equations (3d) ensure that a commodity is not routed on an arc entering

its origin or leaving its destination. Finally inequalities (3e) are the bounds for the x variables and inequalities (3f) are the integer constraints for the design variables.

B.2. Lagrangian Knapsack Relaxation

A standard way to obtain good bounds for the MC problem is to solve the Lagrangian relaxation obtained by dualizing the flow conservation constraints (3b) in formulation (3a)-(3f). Let π_i^k be the Lagrangian multiplier associated with node $i \in N$ and commodity $k \in K$. Dualizing the flow conservation constraints gives the following relaxed Lagrangian problem $LR(\boldsymbol{\pi})$[10]:

$$\min_{(\boldsymbol{x},\boldsymbol{y}) \text{ satisfies (3c)}-\text{(3f)}} \sum_{(i,j)\in A} \left(f_{ij} y_{ij} + \sum_{k\in K} r_{ij}^k x_{ij}^k \right) + \sum_{k\in K} \sum_{i\in N} \pi_i^k \left(b_i^k - \sum_{j\in N_i^+} x_{ij}^k + \sum_{j\in N_i^-} x_{ji}^k \right)$$

Rearranging the terms in the objective function and observing that the relaxed Lagrangian problem is decomposed by arcs, we obtain a subproblem for each arc $(i,j) \in A$ of the form:

$$\begin{aligned}
(LR_{ij}(\boldsymbol{\pi})) \quad & \min_{\boldsymbol{x},\boldsymbol{y}} f_{ij} y_{ij} + \sum_{k\in K_{ij}} w_{ij}^k x_{ij}^k && \text{(4a)}\\
& \sum_{k\in K_{ij}} x_{ij}^k \le c_{ij} y_{ij} && \text{(4b)}\\
& 0 \le x_{ij}^k \le q^k \qquad \forall k \in K_{ij} && \text{(4c)}\\
& y_{ij} \in \{0,1\} && \text{(4d)}
\end{aligned}$$

where $w_{ij}^k = r_{ij}^k - \pi_i^k + \pi_j^k$ and $K_{ij} = \{k \in K \mid j \neq o^k \text{ and } i \neq d^k\}$ is the set of commodities that may be routed through arc (i,j).

For each $(i,j) \in A$, $LR_{ij}(\boldsymbol{\pi})$ is a MILP with only one binary variable. If $y_{ij} = 0$, then, by (4b) and (4c), $x_{ij}^k = 0$ for all $k \in K_{ij}$. If $y_{ij} = 1$, the problem reduces to a continuous knapsack problem. An optimal solution is obtained by ordering the commodities of K_{ij} with respect to decreasing values w_{ij}^k and setting for each variable x_{ij}^k the value $\max\{\min\{q^k, c_{ij} - \sum_{k\in K(k)} q^k\}, 0\}$ where $K(k)$ denotes the set of commodities that preceded k in the order. This step can be done in $O(|K_{ij}|)$ if one computes x_{ij}^k following the computed order. Hence, the complexity of the continuous knapsack problem is $O(|K_{ij}| \log(|K_{ij}|))$. The solution of $LR_{ij}(\boldsymbol{\pi})$ is the minimum between the cost of the continuous knapsack problem and $\mathbf{0}$.

[10] Since the dualized constraints are equations, $\boldsymbol{\pi}$ have no sign constraints.

Lagrangian duality implies that

$$LR(\boldsymbol{\pi}) = \sum_{(i,j)\in A} LR_{ij}(\boldsymbol{\pi}) + \sum_{i\in N} \sum_{k\in K} \pi_i^k b_i^k$$

is a lower bound for the MC problem and the best one is obtained by solving the following Lagrangian dual problem:

$$(LD) \qquad \max_{\boldsymbol{\pi}\in\mathbb{R}^{N\times K}} LR(\boldsymbol{\pi})$$

C. Generalized Assignment Problem

A GA instance is defined by a set I of items and a set J of bins. Each bin j is associated with a certain capacity c_j. For each item $i \in I$ and each bin $j \in J$, p_{ij} is the profit of assigning item i to bin j, and w_{ij} is the weight of item i inside bin j.

Considering a binary variable x_{ij} for each item and each bin that is equal to one if and only if item i is assigned to bin j, the GA problem can be formulated as:

$$\begin{aligned}
& \max_{\boldsymbol{x}} \sum_{i\in I} \sum_{j\in J} p_{ij} x_{ij} && && \text{(5a)}\\
& \sum_{j\in J} x_{ij} \le 1 && \forall i \in I && \text{(5b)}\\
& \sum_{i\in I} w_{ij} x_{ij} \le c_j && \forall j \in J && \text{(5c)}\\
& x_{ij} \in \{0,1\} && \forall i \in I,\ \forall j \in J. && \text{(5d)}
\end{aligned}$$

The objective function (5a) maximizes the total profit. Inequalities (5b) assert that each item is contained in no more than one bin. Inequalities (5c) ensure that the sum of the weights of the items assigned to a bin does not exceed its capacity. Finally, constraints (5d) assure the integrality of the variables.

C.1. Lagrangian Relaxation

A Lagrangian relaxation of the GA problem is obtained by dualizing (5b). For $i \in I$, let $\pi_i \ge 0$ be the Lagrangian multiplier of inequality (5b) associated with item i. For each bin j the subproblem becomes:

$$\begin{aligned}
(LR_j(\boldsymbol{\pi})) \quad & \max_{\boldsymbol{x}} \sum_{i\in I} \sum_{j\in J} (p_{ij} - \pi_i) x_{ij}\\
& \sum_{i\in I} w_{ij} x_{ij} \le c_j\\
& x_{ij} \in \{0,1\} \qquad \forall i \in I
\end{aligned}$$

It corresponds to an integer knapsack with $|I|$ binary variables. For $\boldsymbol{\pi} \ge \mathbf{0}$, the Lagrangian bound $LR(\boldsymbol{\pi})$ is:

$$LR(\boldsymbol{\pi}) = \sum_{j\in J} LR_j(\boldsymbol{\pi}) + \sum_{i\in I} \pi_i.$$

The Lagrangian dual can be then written as:

$$\min_{\boldsymbol{\pi} \in \mathbb{R}^{|I|}_{\geq 0}} LR(\boldsymbol{\pi})$$

D. Dataset collection details

In this appendix, we provide further details on the dataset construction.

Multi-Commodity Fixed-Charge Network Design We generate four datasets of 2000 instances each (1600 for training, 200 for validation and 200 for test) based on Canad instances (Crainic et al., 2001). These canad instances have been chosen such that the Lagrangian dual bound can be solved in nearly one second for the easiest instances and in approximately one hour for the hardest ones.

The first two datasets MC-SML-40 and MC-SML-VAR consider the same graph with 20 nodes and 230 edges, and the same capacity and fixed cost vectors. The first dataset has only instances with 40 commodities whereas the second one has instances with 40, 80, 120, 160 or 200 commodities.

Origins and destinations are randomly chosen using a uniform distribution. Volumes and routing costs are randomly sampled using a Gaussian distribution. Sampling uses four different means μ and variances σ^2 which are determined from the four canad instances p33, p34, p35 and p36 (having the same graph and fixed costs as the datasets) in order to generate four different types of instances: whether the fixed costs are high with respect to routing costs, and whether capacities are high with respect to commodity volumes.

The third dataset MC-BIG-40 is generated similarly as the first one except that it is based on a graph with 30 nodes and 520 edges. The means and variances used to sample the fixed costs and the volumes are determined from the four canad instances p49, p50, p51 and p52. The number of commodities is equal to 40 in each instance.

Finally, the last dataset MC-BIG-VAR contains instances with either the graph, capacities and fixed costs of the first two datasets or the ones of the third dataset. Sampling uses either the canad instances p33, p34, p35 and p36 or the canad instances p49, p50, p51 and p52 for determining the mean and variance, depending on the size of the graph. The number of commodities varies from 40 to 200 if the graph is the one of the first two datasets, and from 40 to 120 otherwise.

Generalized Assignment We create two datasets of GA instances containing 2000 instances each (1600 for training, 200 for validation and 200 for test). The first one contains instances with 10 bins and 100 items whereas the second one contains instances with 20 bins and 400 items. For each dataset, all instances are generated by randomly sampling capacities, weights and profits using a Gaussian distribution of mean μ and variance σ^2 and the values are clipped to an interval $[a, b]$. The values μ, σ^2, a and b are determined from the instance e10100 for the first dataset, and from the instance e20400 for the second one [11]. More specifically, for each type of data (capacities, weights and profits), μ and σ^2 are given by the average and variance of the values of the instance, and a and b are fixed to 0.8 times the minimum value and 1.2 times the maximum value, respectively.

E. Hyperparameters

Model Architecture For all datasets, the MLP F from initial features to high-dimensional is implemented as a linear transformation (8 to 250) followed by a non-linear activation. Then, we consider a linear transformation to the size of the internal representation of nodes for the GNN.

For MC we use 5 blocks, while for GA we use only 3. The fact that for GA are sufficient fewer layers can be explained by looking at the bipartite-graph representation of the instance that is denser for GA than for MC. For instance, in MC, a variable x^k_{ij} appears in three constraints involving several variables while in GA, each variable x_{ij} appears in $|I| + |J|$ constraints so the propagation needs fewer convolutions for the information to be propagated.

The hidden layer of the MLP in the second sub-layer of each block has a size of 1000.

The decoder is an MLP with one hidden layer of 250 nodes.

All non-linear activations are implemented as ReLU. Only the one for the output of the GA is a softplus.

The dropout rate is set to 0.25.

Optimiser Specifications We use as optimizer RAdam, with learning rate 0.0001 for MC and 0.00001 for GA, a Clip Norm (to 5) and exponential decay 0.9, step size 100000 and minimum learning rate 10^{-10}.

GPU specifics For the training on the datasets MC-SML-40, MC-BIG-40, GA-10-100 and GA-20-400 we use GPUs Nvidia Quadro RTX 5000 with 16 GB of RAM. To train the datasets MC-SML-VAR and MC-BIG-VAR we use Nvidia A40 GPUs accelerators with 48Gb of RAM. To test the performance we use Nvidia A40 GPUs accelerators with 48Gb of RAM for all models and all the datasets on validation and test.

CPU specifics The warm starting of the proximal Bundle in SMS++ needs only CPU, the experiments are done on

[11] Instances e10100 and e20400 are GA instances generated by (Yagiura et al., 1999) and available at http://www.al.cm.is.nagoya-u.ac.jp/~yagiura/gap/.

Intel Core i7-8565U CPU @ 1.80GHz × 8.

F. k-NN

We consider the same features as for MLP (see Appendix G) and independently select for each dualized constraint c the 20 nearest neighbors with respect to the Euclidean distance. The LM predicted for c is the mean of the LMs associated with its neighbors. It is important to note that it is a supervised learning method while ours is an unsupervised one. We tried different values of k from 1 to 20 and we find that the best choice is 20. For the implementation we use the julia package NearestNeighbors.jl [12].

G. Features used for MLP and k-NN

Since MLP and k-NN do not use a mechanism such as convolution to propagate the information between the representations of the dualized constraints, we consider for initial features of each dualized constraint all the information provided to our model (see Appendix A for details), as well as a weighted linear combination of variable feature vectors. The weights are the variable coefficients in that constraint and each feature vector contains the initial features provided to our model for the variable and the following additional information:

- the mean values and deviations of the coefficients of that variable on the dualized constraints, and on the non dualized ones,
- its lower and upper bounds.

H. Ablation Study - Number of Layers

In Tables 5 and 6, we present the gaps of three different architectures with varying numbers of layers. The columns represent three different architectures: "Ours," the architecture we introduce in this work; "Nair," the architecture proposed by Nair et al. in (Nair et al., 2020); and "Gasse," the one presented in (Gasse et al., 2019). The rows indicate an incremental number of layers from one to ten.

From Table 6, we observe that for GA, using more than four layers seems to be counterproductive. This can be explained by examining the bipartite graph representation of the instance. In the Generalized Assignment problem, the shortest path between two different nodes associated with the relaxed constraints always consists of four edges. For the Multi-commodity problem, there is no similar bound, as the shortest path between two relaxed nodes depends on the specific structure of the instance. From Table 5, we see that adding more than six layers leads to diminishing improvements, though we can still enhance solution quality by increasing the number of layers.

Gasse's architecture is also more unstable, which could result in significantly higher gaps with some layers compared to fewer layers. This instability may be due to the absence of Layer Normalization, leading to very high gradient values.

Notice that the results in Table 5 and Table 6 show small differences compared to the ones on the main part for 5 layers, as they correspond to other runs of the training.

Table 5. Test set 1 sample - MC-SML-40 - GAP

# Layers	Ours	Nair	Gasse
1	7.56	7.63	9.59
2	5.27	5.23	10.11
3	3.18	3.30	9.47
4	2.62	3.06	2.72
5	2.29	2.47	2.59
6	1.90	2.23	2.76
7	1.80	1.91	2.80
8	1.69	1.76	2.68
9	1.64	1.70	2.84
10	1.56	1.56	3.16

Table 6. Test set 1 sample - GA-10-100 - GAP (s)

# Layers	Ours	Nair	Gasse
1	0.553	0.557	0.70
2	0.543	0.546	0.699
3	0.533	0.524	0.695
4	0.509	0.524	0.690
5	0.512	0.517	0.682
6	0.513	0.518	0.720
7	0.510	0.511	0.785
8	0.512	0.517	0.752
9	0.511	0.515	0.785
10	0.512	0.516	0.722

I. Subgradient Method Initialization

In Table 7 we perform the same experiments as Table 2 with a solver implementing the subgradient method. We see that subgradient method requires much more time than the bundle method, even for low precision levels. The initialization yields more or less the same results. This is likely due to the step size scheduler, which always starts with a step size of one. Then, at a given iteration i, the learning rate is $\frac{1}{1+m}$, where m is the total number of iterations where the predicted value is worse than the previous iteration. An accurate choice of step size can lead to better results, par-

[12] https://juliapackages.com/p/nearestneighbors

Table 7. Impact of initialization for a Sub-Gradient solver on MC-BIG-VAR. We consider initialization from the null vector (zero), the continuous relaxation duals (CR), and our model (Ours). We set the maximum iterations to 100000.

ϵ	zero		CR		Ours	
	time (s)	# iter.	time (s)	# iter.	time (s)	# iter.
1e-1	275.09 ($\pm$ 166.59)	86755.53 ($\pm$ 28222.26)	284.74 ($\pm$ 184.58)	85899.10 ($\pm$ 29126.65)	**271.73** ($\pm$ 168.97)	**84882.55** ($\pm$ 29704.22)
1e-2	281.00 ($\pm$ 168.63)	88175.07 ($\pm$ 27438.57)	291.40 ($\pm$ 186.65)	87513.27 ($\pm$ 28093.75)	**278.55** ($\pm$ 171.83)	**86526.59** ($\pm$ 29003.66)
1e-3	281.00 ($\pm$ 168.64)	88176.12 ($\pm$ 27439.02)	291.66 ($\pm$ 186.67)	87605.24 ($\pm$ 28110.87)	**278.58** ($\pm$ 171.82)	**86540.76** ($\pm$ 29003.68)
1e-4	281.00 ($\pm$ 168.64)	88176.12 ($\pm$ 27439.02)	291.66 ($\pm$ 186.67)	87605.24 ($\pm$ 28110.87)	**278.58** ($\pm$ 171.82)	**86540.76** ($\pm$ 29003.68)

ticularly for non-zero initialization. However, this should be done specifically for each initialization (and possibly for each instance), which is beyond the scope of this work.